Noname manuscript No.
(will be inserted by the editor)

Flexible Nets: A modeling formalism for dynamic systems with uncertain parameters

Jorge Júlvez · Stephen G Oliver

Abstract The modeling of dynamic systems is frequently hampered by a limited knowledge of the system to be modeled and by the difficulty of acquiring accurate data. This often results in a number of uncertain system parameters that are hard to incorporate into a mathematical model. Thus, there is a need for modeling formalisms that can accommodate all available data, even if uncertain, in order to employ them and build useful models. This paper shows how the Flexible Nets (FNs) formalism can be exploited to handle uncertain parameters while offering attractive analysis possibilities. FNs are composed of two nets, an event net and an intensity net, that model the relation between the state and the processes of the system. While the event net captures how the state of the system is updated by the processes in the system, the intensity net models how the speed of such processes is determined by the state of the system. Uncertain parameters are accounted for by sets of inequalities associated with both the event net and the intensity net. FNs are not only demonstrated to be a valuable formalism to cope with system uncertainties, but also to be capable of modeling different system features, such as resource allocation and control actions, in a facile manner.

Keywords Flexible nets; Modeling formalisms; Petri nets; Dynamic systems; Uncertain parameters; Performance analysis.

J. Júlvez (corresponding author)
Cambridge Systems Biology Centre, University of Cambridge, Cambridge, UK
Department of Biochemistry, University of Cambridge, Cambridge, UK
Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain
Tel.: +34 976762336
E-mail: julvez@unizar.es

S. G. Oliver
Cambridge Systems Biology Centre, University of Cambridge, Cambridge, UK
Department of Biochemistry, University of Cambridge, Cambridge, UK

1 Introduction

The development of appropriate models is crucial for the design, analysis and control of dynamic systems. The usefulness of a model depends on both its capacity to capture the relevant features of the system, and its capacity for mathematical analysis. These capacities of the model largely rely on the adopted modeling formalism, i.e. on the set of modeling principles and rules that are used to build the model. The task of modeling is often hindered by the lack of detailed system information.

This paper exploits the particular features of Flexible Nets (FNs), a modeling formalism introduced in [17] to study Wilson disease, to model and analyze dynamic systems with uncertain parameters, to account for partially observable systems and, to compute the control actions that optimize a given control objective. Roughly speaking, a dynamic system can be seen as a set of *state* variables that are modified by means of *processes* (by *process*, we mean any event, operation or activity whose occurrence has the potential to change the state of the system). These two basic entities, *state* and *process*, are mutually related: on the one hand, the execution of the processes determines how the state changes; on the other hand, the state determines the speed of the processes. These relationships between state and processes are clear, for instance, in a chemical system where the state is given by the amount of molecules and the processes are the reactions taking place in the system. On the one hand, the occurrence of reactions produces a change in the amount of molecules that satisfies the stoichiometry of these reactions. On the other hand, the rate of the reactions depends on the amount of molecules. FNs capture these relationships between state and processes by means of two different nets: the event net and the intensity net.

Let us introduce some of the basic features of FNs by means of a simple chemical reaction network. Assume that the reaction network is composed of the following two reactions:

$$R_1 : \emptyset \rightarrow A \qquad\qquad R_2 : A \rightarrow nB + 4nC$$

Reaction R_1 models the production of compound A (each occurrence of R_1 increases the concentration of A, which is denoted $[A]$, one unit), and reaction R_2 models how A is decomposed into compounds B and C. The amounts which $[B]$ and $[C]$ are increased by the occurrence of R_2 depend on n. Let us assume that n is uncertain, but known to be in the interval $[20, 22]$. That is, each occurrence of R_2 decreases $[A]$ one unit, increases $[B]$ n units, and increases $[C]$ $4n$ units where $n \in [20, 22]$. Let us further assume that the initial concentrations of $[B]$ and $[C]$ are 0, and the initial concentration of $[A]$ is known to be in the interval $[9.9, 1.1]$.

The FN in Figure 1 models the described reaction network. Namely, each chemical compound is a associated with a circle (which will be called a place), and each reaction is associated with a rectangle (which will be called a transition). The stoichiometry of the reactions is modeled by the equations associated with the dots labelled v_1 and v_2 (which will be called event handlers). The uncertain stoichiometry of R_2 is accounted for by the inequalities associated with v_2, i.e. $a{=}v$, $20v{\leq}b{\leq}22v$ and $c{=}4b$; and the uncertain initial concentration $[A]$ is captured by the inequalities associated with A, i.e. $9.9{\leq}m_0[A]{\leq}1.1$.

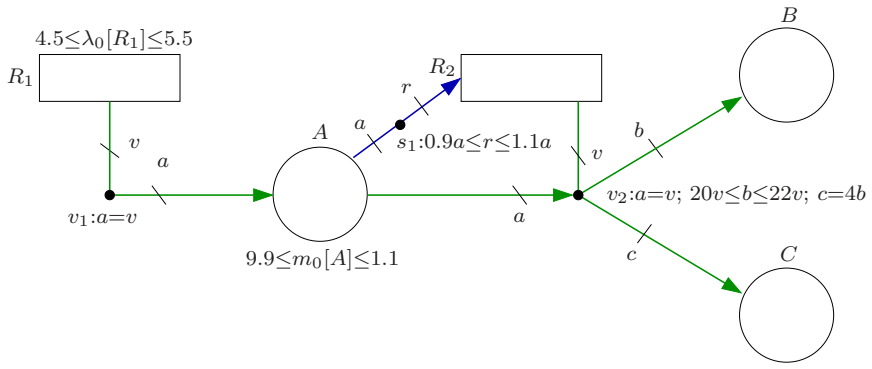


Fig. 1 FN modeling a simple chemical reaction network.

The event net of an FN is a graph with three different types of vertices: *places*, *transitions* and *event handlers*. While places and transitions are used to model the state and the processes of the system, respectively, event handlers are used to determine the quantities by which the state is changed when given processes occur. Each place is associated with a state variable, and the value of that variable at a given instant is called marking, or number of *tokens*, in that place. Similarly, each transition is associated with a process, and the number of times the process has taken place is the number of *actions* in the transition. Each event handler connects a set of transitions to a set of places, and is associated with a set of linear inequalities that relates actions to marking. Given a number of actions in the connected transitions, any solution of the set of linear inequalities can be used to update the number of tokens in the connected places. Thus, the amount by which the marking changes is allowed to be nondeterministic. This feature of event nets allows the model to account for the different system evolutions that can arise as a consequence of uncertainty in the system. In Figure 1, the event net is composed of the places, transitions, event handlers and arcs and edges in green.

Let us further assume that the rate of reaction R_1 is uncertain, but constrained to the interval $[4.5, 5.5]$, i.e. the number of reactions that occur per time unit is in $[4.5, 5.5]$, and the rate R_2 satisfies $0.9[A] \leq rate(R_2) \leq 1.1[A]$, i.e. it is proportional to $[A]$ with an uncertainty of 10%. These reaction rates are modeled in the FN by the inequality associated with R_1 and the inequality associated with the dot labelled s_1 (which will be denoted intensity handler).

Similarly to event nets, an intensity net is a graph with three different types of vertices: *places*, *transitions* and *intensity handlers*. Places and transitions have the same role as in event nets. Each intensity handler connects a set of places to a set of transitions, and is associated with a set of linear inequalities that relates the number of tokens with the speed of the transitions, i.e. of the processes modeled by the transitions. The intensity, or speed, of a transition determines the rate at which actions are created in the transition. As in event nets, any solution of the inequalities can be used to determine the speed of transition, thus, linear inequalities can be used to model uncertainties in the dynamics of the system. In Figure 1, the intensity net is composed by the places, transitions, intensity handlers and arcs and edges in blue.

An FN is the result of combining an event net and an intensity net, i.e. an FN is a graph with four types of vertices: *places*, *transitions*, *event handlers* and *intensity handlers*. While the intensity net establishes the speeds at which actions are generated as a function of the marking, the event net specifies how the generated actions are executed and how a new marking is computed. Thus, although FNs are inspired by Petri nets [21], their structure is different, since in addition to *places* and *transitions*, FNs have *handlers* which are connected to places and transitions by *arcs* and *edges*. FNs offer both high modeling power and appealing analysis possibilities that aim to make use of all the information provided by the available uncertain parameters. Namely, FNs can accommodate uncertainties in the initial marking, in the marking change produced by the firing of the transitions, in the default speeds of transitions, and in the speed of transitions produced by the marking.

A number of different formalisms can be found in the literature that can, to some extend, incorporate uncertain parameters in their models. Depending on the domain of their state variables, these formalisms can be roughly classified as those whose variables are integer numbers, and those whose variables are real numbers (hybrid approaches combine both types of variables).

In the discrete domain, extensions of the most popular modeling formalisms exist that can handle uncertain parameters. For instance, in the Petri nets arena, uncertain knowledge of the marking is particularly well handled by fuzzy [19] and possibilistic Petri nets [9], uncertainty in the firing of transitions can be accounted for by labeled [8] and interpreted [25] Petri nets, and uncertain firing times can be modeled by time Petri nets [20] and stochastic Petri nets [2]. Other discrete formalisms that include extensions to account for uncertain parameters are probabilistic Boolean networks [26], which are an extension of Boolean networks [32], and influence diagrams [12], which are generalizations of Bayesian networks [22] that can solve decision problems under uncertainty. Stochastic extensions of timed automata [3] also exist in which delays and discrete choices are made randomly [4]. In a similar vein, stochastic extensions of process algebras [24, 10] have been proposed to describe components with uncertain behaviour [11].

A major difference of the above mentioned approaches with respect to FNs is that the state variables of FNs are real numbers. This implies that genuinely discrete systems cannot be modeled by FNs. Nevertheless, the use of real variables facilitates, in general, the use of more efficient analysis methods since the state explosion problem inherent to large discrete systems is avoided, and linear programming techniques can be applied. Moreover, large discrete populations can be approximated reasonably well in many cases by means of real variables [5]. With respect to the existing stochastic approaches and extensions, it should be said that they offer the possibility to perform useful statistical analyses, which usually require information about the probability distributions of the system, and often involve a significant computational cost. In contrast, FNs do not require information about probability distributions, just about the intervals in which the uncertain parameters lay. This results in efficient analysis techniques based on linear programming.

The most popular modeling approaches in the continuous domain are based on differential equations [7, 29, 30]. In particular, the relaxation of the integrality constraint in a discrete formalism usually leads to models that are governed by dif-

ferential equations. For instance, the evolution of continuous Petri nets [27], which can be seen as a relaxation of Petri nets [21], is determined by a set of ordinary differential equations. Another popular modeling formalism that can graphically represent systems in different domains and that can be easily converted to state space representation is bond graphs [6]. However, it should be emphasised that a potential difficulty in the design of models based on differential equations is that exhaustive and accurate information about the system dynamics is required, i.e. uncertain parameters cannot be easily handled. Note that the time trajectory of a system modeled by ordinary differential equations is continuous and deterministic. On the other hand, constraint-based models, which are popular in systems biology [31, 23], can incorporate uncertain dynamic information but their analysis capabilities are limited to the steady state.

An important feature of FNs is that they can bridge the gap between deterministic models and constraint-based models by allowing the incorporation of uncertain parameters. Thus, on the one hand and similarly to continuous Petri nets which are deterministic [16], FNs can model positive linear systems; on the other hand and similarly to constraint-based models [31], FNs can model systems with uncertain initial state and uncertain process speeds. In contrast to Petri nets, the timing of transitions and the marking changes in places are explicitly separated in FNs: the timing is handled by the intensity handlers, and the marking changes by the event handlers. This can lead to a clearer and more concise graphical representation of the system. Moreover, efficient computational methods exist to analyse the transient state of FNs.

The rest of the paper is organized as follows: Section 2 introduces event nets and shows how a partially observable system can be modeled. Intensity nets are presented in Section 3. The combination of these nets leads to FNs, which are defined in Section 4. Section 5 shows how FNs can handle systems with uncertain parameters and analyze them. Section 6 concludes the paper.

2 Event nets

In the following, the reader is assumed to be familiar with Petri nets (see [21] for a gentle introduction).

2.1 Definition and state equations

This section introduces event nets, which can be denoted as TVP nets, i.e. actions in transitions T produce and consume tokens in places P through event handlers V. Event handlers connect places and transitions, and determine the marking changes according to the actions in the transitions. In contrast to Petri nets, the net elements that produce changes in the marking are the event handlers, and such changes are allowed to be nondeterministic.

Definition 1 (Event net) *An event net is a tuple* $\mathcal{N}_V = (P, T, V, E_V, A, B)$ *where* (P, T, V, E_V) *is a tripartite graph determining the net structure and* (A, B) *are matrices determining the potential evolutions of the marking.*

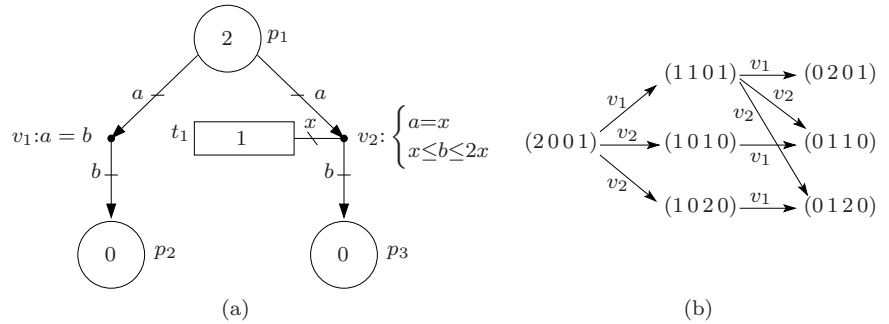


Fig. 2 (a) Event net. (b) Potential state evolutions with discrete firings of the event handlers (the components of the vectors correspond to the variables $(m[p_1]\ m[p_2]\ m[p_3]\ a_T[t_1])$).

The set of vertices of the net is partitioned into three sets:

- $P = \{p_1, \ldots, p_i, \ldots\}$ is a set of $|P|$ places.
- $T = \{t_1, \ldots, t_j, \ldots\}$ is a set of $|T|$ transitions.
- $V = \{v_1, \ldots, v_k, \ldots\}$ is a set of $|V|$ event handlers.

The places, depicted as circles, model the different types of components or elements in the system, e.g. resources, products, items, etc. The transitions, depicted as rectangles, model the different types of operations, activities or processes in the system. Such operations require time to be performed and have the potential to change, i.e. produce and consume, the amount of components, i.e. the marking. The event handlers, depicted as dots, model the different ways in which the transitions can change the marking.

The vertices of the net are connected by the edges in E_V. Each pair of vertices can be connected by at most one edge. The set E_V is partitioned into two sets E_V^P and E_V^T, where E_V^P is a set of *directed edges* connecting places to event handlers and vice versa, and E_V^T is a set of *undirected edges* connecting transitions and event handlers. For simplicity, directed edges are referred as *arcs*, and undirected edges as *edges*. More formally:

- Every $e \in E_V^P$ is either an arc $e = (p_i, v_k)$ from a place p_i to a handler v_k, or an arc $e = (v_k, p_i)$ from a handler v_k to a place p_i.
- Every $e \in E_V^T$ is an edge $e = \{t_j, v_k\}$ connecting a transition t_j and a handler v_k.

Notice that direct connections among places and transitions are not allowed. The following notation is used:

- ${}^p v_k$ denotes the input places of v_k, i.e. ${}^p v_k = \{p_i | (p_i, v_k) \in E_V^P\}$
- v_k^p denotes the output places of v_k, i.e. $v_k^p = \{p_i | (v_k, p_i) \in E_V^P\}$
- ${}^v p_i$ denotes the input handlers of p_i, i.e. ${}^v p_i = \{v_k | (v_k, p_i) \in E_V^P\}$
- p_i^v denotes the output handlers of p_i, i.e. $p_i^v = \{v_k | (p_i, v_k) \in E_V^P\}$
- ${}^t v_k$ denotes the transitions connected to v_k, i.e. ${}^t v_k = \{t_j | \{t_j, v_k\} \in E_V^T\}$
- t_j^v denotes the handlers connected to t_j, i.e. $t_j^v = \{v_k | \{t_j, v_k\} \in E_V^T\}$

Example 1 The event net in Figure 2(a) has three places, $P = \{p_1, p_2, p_3\}$, one transition, $T = \{t_1\}$, and two event handlers $V = \{v_1, v_2\}$. The set of arcs is

$E_V^P = \{(p_1, v_1), (v_1, p_2), (p_1, v_2), (v_2, p_3)\}$, and the set of edges is $E_V^T = \{\{t_1, v_2\}\}$. The set of all arcs and edges is $E_V = E_V^P \cup E_V^T$. As examples for the introduced notation, the set of output handlers of p_1 is $p_1^v = \{v_1, v_2\}$, the set of transitions connected to v_2 is ${}^t v_2 = \{t_1\}$, and the set of input handlers of p_3 is ${}^v p_3 = \{v_2\}$.

In an event net, each place contains a number of tokens (or marking), and each transition contains a number of actions that represent the potential of the system to carry out the associated process. In contrast to tokens, actions require time to be produced (the production rate of actions is determined by the intensity net, see Section 3). The state of an event net accounts not only for the marking and the number of actions, but also for the marking changes and the execution of actions:

Definition 2 (State) *The state of an event net $\mathcal{N}_V$ is given by the tuple $(\sigma, a_T, a_E, \Delta m, m)$, where:*

- $\sigma \in \mathbb{R}_{\geq 0}^{|T|}$ *is a vector indexed by T where $\sigma[t_j]$ is the number of actions produced in t_j.*
- $a_T \in \mathbb{R}_{\geq 0}^{|T|}$ *is a vector indexed by T where $a_T[t_j]$ is the number of actions available in t_j.*
- $a_E \in \mathbb{R}_{\geq 0}^{|E_V^T|}$ *is a vector indexed by E_V^T where $a_E[\{t_j, v_k\}]$ is the number of actions of t_j executed by v_k.*
- $\Delta m \in \mathbb{R}_{\geq 0}^{|E_V^P|}$ *is a vector indexed by E_V^P where $\Delta m[(p_i, v_k)]$ is the number of tokens in p_i consumed by v_k, and $\Delta m[(v_k, p_i)]$ is the number of tokens in p_i produced by v_k.*
- $m \in \mathbb{R}_{\geq 0}^{|P|}$ *is the marking, i.e. a vector indexed by P where $m[p_i]$ is the number of tokens in p_i.*

Thus, every net element (except for the event handlers) is associated with at least one nonnegative real variable. Since actions need time to be produced, at the initial state it holds $\sigma = 0$, $a_T = 0$ and $a_E = 0$. Notice that dealing with real state variables instead of discrete ones allows the model to incorporate real quantities and to approximate large discrete quantities as in continuous Petri nets [27]. In any case, the state variables can be constrained to the nonnegative integers if required, see Subsection 2.2.

Each event handler $v_k \in V$ is associated with a set of linear inequalities that relate the number of actions executed in the connected transitions to the marking changes in the connected places. The coefficients of such a set of inequalities can be expressed by two matrices (A_k, B_k) of real numbers and the same number of rows that are associated with each handler $v_k \in V$. The number of actions executed by v_k, a_f, and the produced marking changes, Δm_f, is given by $A_k \Delta m_f \leq B_k a_f$. The columns of A_k are indexed by the arcs connecting v_k to places. The columns of B_k are indexed by the edges connecting transitions to v_k. Matrix $A(B)$ is obtained by arranging all the matrices $A_k(B_k)$ *diagonally.*

Example 2 The inequalities associated with the event handlers of the net in Figure 2(a) are: v_1:a=b and v_2: $\begin{cases} a=x \\ x \leq b \leq 2x \end{cases}$, where '$a$', '$b$' and '$x$' are used to label arcs and edges. More precisely, in v_1:a=b, 'a' denotes the number of tokens in p_1

consumed by v_1, and 'b' denotes the number of tokens in p_2 produced by v_1. In the inequalities associated with v_2, 'a' and 'b' denote the number of tokens in p_1 consumed by v_2 and the number of tokens in p_3 in produced by v_2 respectively, and 'x' denotes the number of actions in t_1 executed by v_2. The equality v_1:a=b means that the number of tokens in p_1 consumed by v_1 is equal to the number of tokens in p_2 produced by v_1. In other words, for every token in p_1 consumed by v_1, a token is produced in p_2 by v_1, this can be interpreted as tokens moving from p_1 to p_2 through v_1.

The equation a=x associated with v_2 means that the number of tokens in p_1 consumed by v_2 is equal to the number of actions in t_1 executed by v_2, e.g. if one action is executed then one token is consumed. Moreover, the inequality $x \leq b \leq 2x$ means that the execution of one action in t_1 by v_2, i.e. $x = 1$, produces a nondeterministic quantity $b \in [1, 2]$ of tokens in p_3 (each execution of an action can produce a different amount $b \in [1, 2]$ of tokens in p_3).

Notice that v_1 is not connected to any transition. This means that no process is required to move a token from p_1 to p_2. For the sake of mathematical notation, it can be assumed that v_1 is connected to a fake transition, t_{fake}, that has no effect on the model. The matrices A_1, B_1, A_2 and B_2 that capture the inequalities associated with the event handlers are:

$$A_1 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}; B_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; A_2 = \begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix}; B_2 = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 2 \end{pmatrix}$$

where the indices of the columns of A_1 are ordered as (p_1, v_1), (v_1, p_2); the index of the column of B_1 is $\{t_{fake}, v_1\}$; the indices of the columns of A_2 are ordered as (p_1, v_2), (v_2, p_3); and the index of B_2 is $\{t_1, v_2\}$. Thus, the number of actions executed and marking changes produced by v_2 are related by:

$$\begin{pmatrix} 1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta m[(p_1, v_2)] \\ \Delta m[(v_2, p_3)] \end{pmatrix} \leq \begin{pmatrix} 1 \\ -1 \\ -1 \\ 2 \end{pmatrix} (a_E[\{t_1, v_2\}])$$

Matrices A_1 and A_2(B_1 and B_2) can be arranged diagonally to obtain $A(B)$:

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \quad B = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & -1 \\ 0 & -1 \\ 0 & 2 \end{pmatrix}$$

where the indices of the columns of A are (p_1, v_1), (v_1, p_2), (p_1, v_2), (v_2, p_3); and the indices of the columns of B are $\{t_{fake}, v_1\}$, $\{t_1, v_2\}$.

Notice that the number of actions produced in a transition (by the intensity net), σ, is equal to the number of actions that have been executed by the connected event handlers, a_E, plus the number of actions still available, a_T, hence, it holds:

$$\sigma[t_j] = a_T[t_j] + \sum_{v_k \in t_j^v} a_E[\{t_j, v_k\}] \quad \forall\, t_j \in T \tag{1}$$

Similarly, the number of tokens in a place p_i is equal to the initial number of tokens, which is denoted $m_0[p_i]$, minus the number of tokens consumed plus the

number of tokens produced by the connected event handlers:

$$m[p_i]{=}m_0[p_i]{-}\sum_{v_k\in p_i^v}\Delta m[(p_i,v_k)]{+}\sum_{v_k\in {}^v p_i}\Delta m[(v_k,p_i)]\quad\forall\ p_i\in P \tag{2}$$

The event net establishes how the state evolves as event handlers are enabled and fire.

Definition 3 (Enabling) *Event handler v_k is enabled at $(\sigma, a_T, a_E, \Delta m, m)$ if a vector $a_f \in \mathbb{R}_{\geq 0}^{|{}^t v_k|}$ indexed by the edges of v_k, and a vector $\Delta m_f \in \mathbb{R}_{\geq 0}^{|{}^p v_k|+|v_k^p|}$ indexed by the arcs of v_k exist such that:*

$$a_f[\{t_j,v_k\}] \leq a_T[t_j] \quad \forall\ t_j \in {}^t v_k \tag{3}$$
$$A_k \Delta m_f \leq B_k a_f \tag{4}$$
$$\Delta m_f[(p_i,v_k)] \leq m[p_i] \quad \forall\ p_i \in {}^p v_k \tag{5}$$
$$\mathbf{1}a_f + \mathbf{1}\Delta m_f > 0 \tag{6}$$

Inequality (3) guarantees that enough actions are available, (4) makes use of the matrices A_k and B_k to relate the number of executed actions to the marking changes, (5) guarantees that enough tokens are available in the input places to be consumed, (6) guarantees that the overall state change is not null. Notice that the inequalities (4) allow the modeling of uncertainty in the marking changes produced by the execution of actions.

Definition 4 (Firing) *An event handler v_k enabled at $(\sigma, a_T, a_E, \Delta m, m)$ can fire. The firing of v_k leads instantaneously to a new state $(\sigma, a'_T, a'_E, \Delta m', m')$ where only the variables associated with edges, arcs, places and transitions connected to v_k are updated as follows:*

$$\begin{aligned}
a'_T[t_j] &= a_T[t_j] - a_f[\{t_j,v_k\}] && \forall\ t_j{\in}{}^t v_k\\
a'_E[\{t_j,v_k\}] &= a_E[\{t_j,v_k\}] + a_f[\{t_j,v_k\}] && \forall\ t_j{\in}{}^t v_k\\
\Delta m'[(p_i,v_k)] &= \Delta m[(p_i,v_k)] + \Delta m_f[(p_i,v_k)] && \forall\ p_i{\in}{}^p v_k\\
\Delta m'[(v_k,p_i)] &= \Delta m[(v_k,p_i)] + \Delta m_f[(v_k,p_i)] && \forall\ p_i{\in} v_k^p\\
m'[p_i] &= m[p_i]{-}\Delta m_f[(p_i,v_k)] && \forall\ p_i{\in}{}^p v_k\\
m'[p_i] &= m[p_i]{+}\Delta m_f[(v_k,p_i)] && \forall\ p_i{\in} v_k^p
\end{aligned}$$

where a_f and Δm_f satisfy (3), (4), (5) *and* (6).

Notice that an enabled handler is not forced to fire, and that the state reached by the firing of an event handler is allowed to be nondeterministic (see inequality (4)). Moreover, the firing does not force the execution of a minimum number of actions nor the consumption or production of a minimum number of tokens. In fact, the equations in Definition 4 are trivially satisfied with $a_f = 0$ and $\Delta m_f = 0$. Thus, such equations also hold for every non-enabled handler with $a_f = 0$ and $\Delta m_f = 0$.

The overall change in the state produced by several firings is the result of adding the changes produced by each firing. This leads to a set of equations that are satisfied by the states that can be reached from the initial state.

Proposition 1 (State equations) *Let the state of an event net $\mathcal{N}_V$ be $(\sigma, \sigma, 0, 0, m_0)$, i.e. σ actions are available and no event handler has fired. Every state $(\sigma, a_T, a_E, \Delta m, m)$ reachable from $(\sigma, \sigma, 0, 0, m_0)$ belongs to $SE_{\mathcal{N}_V}(\sigma, m_0)$ where:*

$$\begin{aligned} SE_{\mathcal{N}_V}(\sigma, m_0) = \{(\sigma, a_T, a_E, \Delta m, m) | \\ \sigma = a_T + Y_\sigma a_E \\ A\Delta m \leq B a_E \\ m = m_0 + Z_m \Delta m\} \end{aligned} \tag{7}$$

where Y_σ and Z_m are determined by the net structure:

- *Y_σ is a matrix with rows indexed by T, columns indexed by E_V^T, and such that $Y_\sigma[t_j, \{t_j, v_k\}] = 1 \ \forall \ \{t_j, v_k\} \in E_V^T$ and the rest of the elements in Y_σ are 0,*
- *Z_m is a matrix with rows indexed by P, columns indexed by E_V^P, and such that $Z_m[p_i, (p_i, v_k)] = -1 \ \forall \ (p_i, v_k) \in E_V^P$, $Z_m[p_i, (v_k, p_i)] = 1 \ \forall \ (v_k, p_i) \in E_V^P$ and the rest of the elements in Z_m are 0,*

and a_T, a_E, Δm and m are nonnegative variables.

Proof Let us show that equations (7) necessarily hold for every state $(\sigma, a_T, a_E, \Delta m, m)$ reachable from $(\sigma, \sigma, 0, 0, m_0)$. Equation $\sigma = a_T + Y_\sigma a_E$ in (7) states that the number of actions produced, σ, is equal to the number of actions executed, a_E, plus the number of actions available, a_T. This is equivalent to (1) expressed in matrix form, and thus necessarily holds. Equation $A\Delta m \leq B a_E$ in (7) is the matrix form of (4) and accounts for all the actions executed and all the marking changes produced by the firing of all the event handlers. That is, it captures all the cumulative marking changes, Δm, produced by all the executed actions, a_E, and, hence, must necessarily hold. Finally, Equation $m = m_0 + Z_m \Delta m$ in (7) which updates the number of tokens, m, in all the places according to the cumulative marking changes, Δm, is the matrix form of (2) and, hence, must also hold. □

Roughly speaking, the role of matrix Y_σ is to distribute the actions in transitions among the handlers connected to them, see (1). The role of Z_m is to collect and add the marking changes produced by the firings, see (2).

Example 3 Equation $\sigma = a_T + Y_\sigma a_E$ for the event net in Figure 2(a) assuming again that v_1 is connected to a fake transition is:

$$\begin{pmatrix} \sigma[t_1] \\ \sigma[t_{fake}] \end{pmatrix} = \begin{pmatrix} a_T[t_1] \\ a_T[t_{fake}] \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} a_E[\{t_1, v_2\}] \\ a_E[\{t_{fake}, v_1\}] \end{pmatrix}$$

and equation $m = m_0 + Z_m \Delta m$ is:

$$\begin{pmatrix} m[p_1] \\ m[p_2] \\ m[p_3] \end{pmatrix} = \begin{pmatrix} m_0[p_1] \\ m_0[p_2] \\ m_0[p_3] \end{pmatrix} + \begin{pmatrix} -1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta m[(p_1, v_1)] \\ \Delta m[(v_1, p_2)] \\ \Delta m[(p_1, v_2)] \\ \Delta m[(v_2, p_3)] \end{pmatrix}$$

Let us assume that the marking of the net in Figure 2(a) is $m[p_1]$=2, $m[p_2]$=0, $m[p_3]$=0, that one action was produced in t_1, i.e. $\sigma[t_1]$=1, and it is available, i.e, $a_T[t_1]$=1, and no event handler has fired. This corresponds to the state ($\sigma[t_1]$=1, $a_T[t_1]$=1, $a_E[\{t_1, v_2\}]$=0, $(\Delta m[(p_1, v_1)]$=0, $\Delta m[(v_1, p_2)]$=0, $\Delta m[(p_1, v_2)]$=0,

$\Delta m[(v_2, p_3)]$=0), $(m[p_1]$=2, $m[p_2]$=0, $m[p_3]$=0)). At this state, both event handlers, v_1 and v_2, are enabled and can fire. If v_2 fires in an amount of 1, i.e. x=1, then: $a_T[t_1]$ and $m[p_1]$ are decreased by 1; $a_E[\{t_1, v_2\}]$ and $\Delta m[(p_1, v_2)]$ are increased by one; and $\Delta m[(v_2, p_3)]$ and $m[p_3]$ are increased by a nondeterministic quantity in the interval $[1, 2]$ (the value of $\sigma[t_1]$ remains unaltered as no actions are produced).

The graph in Figure 2(b) shows the potential evolutions of the net under the assumption that event handlers fire in discrete amounts, i.e. all markings and actions are integers. The components of the vectors of states in the graph correspond to the variables ($m[p_1]$ $m[p_2]$ $m[p_3]$ $a_T[t_1]$). The arcs are labeled with the event handler that is fired. Remark that while the firing of v_1 produces a deterministic change (one token consumed from p_1 and one token produced in p_2), the state change produced by the firing of v_2 is nondeterministic (either one or two tokens can be produced in p_3).

Notice that equations (7) account for the cumulative effect, and not the sequence, of the firings. In particular, the availability of tokens and actions consumed by the sequence of firings is not checked. This can lead to spurious solutions [28] in the state equations. Hence, equations (7) represent a necessary condition for the reachability of $(\sigma, a_T, a_E, \Delta m, m)$.

In order to account for linear relationships among the values of the initial marking, m_0 is assumed to be a vector constrained as:

$$J_m m_0 \leq K_m \tag{8}$$

where J_m and K_m are real matrices of appropriate size. Note that the inequalities (8) can be used to account for the uncertain initial marking of a place, e.g. $10 \leq m_0[p_1] \leq 12$, or to express linear constraints among markings, e.g. $m_0[p_1] + m_0[p_2] = 5$ and $m_0[p_4] = 2m_0[p_3]$. Equations (7) can be easily modified to take into account the relationships expressed by (8):

$$\begin{aligned} SE_{\mathcal{N}_V}(\sigma, J_m, K_m) = \{(\sigma, a_T, a_E, \Delta m, m) | \\ \sigma = a_T + Y_\sigma a_E \\ A\Delta m \leq B a_E \\ m = m_0 + Z_m \Delta m \\ J_m m_0 \leq K_m\} \end{aligned} \tag{9}$$

Although event handlers are not forced to fire, it is useful in some cases to consider only those states in which all the actions of given transitions have been executed. Let $T_F \subseteq T$ be the set of transitions whose actions must have been executed, i.e. the number of available actions of $t_j \in T_F$ must be 0. In order to constrain (9) to such a set of states, the following equation can be added:

$$a_T[t_j] = 0 \quad \forall\, t_j \in T_F \tag{10}$$

2.2 Partial observability

In an event net, the marking change produced by the execution of an action is allowed to be nondeterministic. This is the case when there are several event handlers connected to a transition, or if the connected event handler has appropriate

inequalities associated with it. The nondeterministic effect of the execution of actions can be used to develop nondeterministic models and, in particular, to model partially observable systems. The state equations (9) account for all the states that can be reached after the firing of event handlers. In the context of partial observability, these equations characterize the set of states that are consistent with a given observation of a partially observable system. The following example shows how a partially observable system can be modeled by an event net.

For the sake of this example, let us assume that the event handlers of the net in Figure 3(a) only fire in discrete amounts and that only the sequence of transitions whose actions are executed is observable. For clarity, the equations of event handlers that make all their labels equal are omitted, e.g. the equations of v_1 are $a{=}b$ and $a{=}x$, and they are therefore omitted (this same omission is made in the nets of the rest of the paper).

Thus, the observation of t_1 corresponds to the firing of v_1; the firing of v_2 is unobservable (silent event) because it is not connected to a transition; the observation of t_2 corresponds either to the firing of v_3 or v_4 because both handlers use the actions in t_2 (in other words, t_2 models events that cannot be distinguished by an observer); the observation of t_3 corresponds to the firing of v_5.

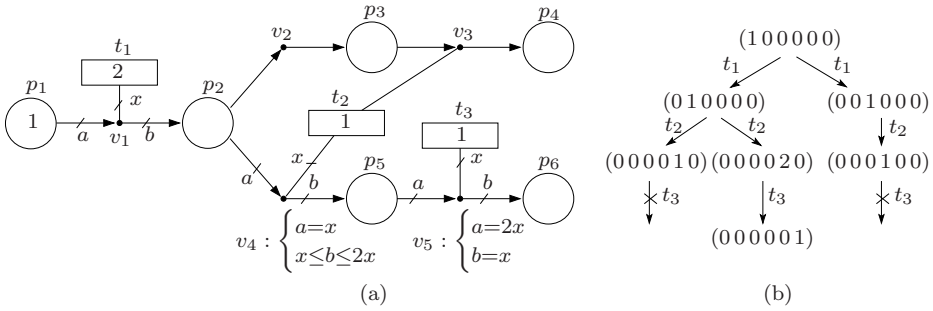


Fig. 3 (a) Event net modeling a partially observable system. (b) Potential evolutions of the marking with discrete firings.

The graph in Figure 3(b) shows the potential evolutions of the net with $m_0 = (1\,0\,0\,0\,0\,0)$ and $\sigma = (2\,1\,1)$, the components of the vectors in the graph correspond to the marking of places $(p_1\,p_2\,p_3\,p_4\,p_5\,p_6)$. Initially, only v_1 can fire and, hence, the execution of actions in t_1 is the only event that can be observed. The observation of t_1 means that a token was consumed from p_1 and a token was produced in p_2. Since p_2 is the input place of v_2 whose firing cannot be observed, a token in p_2 can remain in p_2 or *move silently* to p_3. Thus, the set of markings consistent with the observation of t_1 is $\{(0\ 1\ 0\ 0\ 0\ 0), (0\ 0\ 1\ 0\ 0\ 0)\}$, see second row of the graph. Assume that t_2 is now observed, i.e. either v_3 or v_4 has fired. If the marking was $(0\ 0\ 1\ 0\ 0\ 0)$, then v_3 (the only enabled handler) has fired and that leads to marking $(0\ 0\ 0\ 1\ 0\ 0)$. Otherwise, v_4 has fired and that leads either to $(0\ 0\ 0\ 0\ 1\ 0)$ or $(0\ 0\ 0\ 0\ 2\ 0)$ because one firing of v_4 can produce one or two tokens in p_5. Thus, the set of markings consistent with the observation of the sequence of events t_1 and t_2 is $\{(0\ 0\ 0\ 0\ 1\ 0), (0\ 0\ 0\ 0\ 2\ 0), (0\ 0\ 0\ 1\ 0\ 0)\}$, see third row of the graph. Assume that t_3 is now observed, i.e. v_5 has fired. Notice that the

firing of v_5 requires two tokens in p_5, thus, after the observation of t_1 and t_2, (0 0 0 0 2 0) is the only consistent marking at which v_5 can fire. Consequently, the only sequence of markings consistent with the observation of t_1, t_2 and t_3 is (1 0 0 0 0 0), (0 1 0 0 0 0), (0 0 0 0 2 0) and (0 0 0 0 0 1).

Notice that the state equations (7) can be used straightforwardly to compute the set of consistent markings with a given observation. For instance, for the observation of t_1, t_2 and t_3 discussed above, the marking $m = (0\ 0\ 0\ 0\ 0\ 1)$ is the only solution of:

$$\sigma = a_T + Y_\sigma a_E;\ A\Delta m \leq Ba_E;\ m = m_0 + Z_m \Delta m;\ \sigma = (2\ 1\ 1);\ a_T = (1\ 0\ 0)$$

where $\sigma = (2\ 1\ 1)$ and $a_T = (1\ 0\ 0)$ are the number of actions available at the beginning and at the end respectively. That is, each transition was observed once, i.e. $\sigma - a_T = (1\ 1\ 1)$.

3 Intensity nets

3.1 Definition and state equations

This section introduces intensity nets, which can be denoted as PST nets, i.e. tokens in places P produce and consume intensities in transitions T through intensity handlers S. The intensity of a transition t_j is the speed at which actions are produced in t_j. In other words, the number of actions produced at t is given by the integral over time of the intensity of t_j. Intensity nets and event nets operate in a similar fashion. In fact, the changes in the intensities are produced by tokens in the intensity net in the same way that changes in the marking are produced by actions in the event net.

Definition 5 (Intensity net) *An intensity net is a tuple $\mathcal{N}_S = (P, T, S, E_S, C, D)$ where (P, T, S, E_S) is a tripartite graph determining the net structure and (C, D) are matrices determining the potential intensity changes produced by the marking.*

The set of vertices of the net is partitioned into three sets, P is the set of places, T is the set of transitions, and:

- $S = \{s_1, \ldots, s_l, \ldots\}$ is a set of $|S|$ intensity handlers.

Places and transitions model the same system features as in the event net. The intensity handlers are depicted as dots and model the different ways in which the tokens can generate intensities in the transitions.

The vertices of the net are connected by the edges in E_S. Each pair of vertices can be connected by at most one edge. The set E_S is partitioned into two sets E_S^T and E_S^P, where E_S^T is a set of directed edges (or simply arcs) connecting transitions to intensity handlers and vice versa, and E_S^P is a set of undirected edges (or simply edges) connecting places and intensity handlers. Thus, although both event handlers and intensity handlers are represented as dots, they can be easily distinguished by the arcs and edges that connect them to transitions and places. More formally:

- Every $e \in E_S^T$ is either an arc $e = (t_j, s_l)$ from a transition t_j to a handler s_l, or an arc $e = (s_l, t_j)$ from a handler s_l to a transition t_j.

– Every $e \in E_S^P$ is an edge $e = \{p_i, s_l\}$ connecting a place p_i and a handler s_l.

As in the event net, connections among places and transitions are not allowed. The following notation is used:

– ${}^t s_l$ denotes the input transitions of s_l, i.e. ${}^t s_l = \{t_j | (t_j, s_l) \in E_S^T\}$
– s_l^t denotes the output transitions of s_l, i.e. $s_l^t = \{t_j | (s_l, t_j) \in E_S^T\}$
– ${}^s t_j$ denotes the input handlers of t_j, i.e. ${}^s t_j = \{s_l | (s_l, t_j) \in E_S^T\}$
– t_j^s denotes the output handlers of t_j, i.e. $t_j^s = \{s_l | (t_j, s_l) \in E_S^T\}$
– ${}^p s_l$ denotes the places connected to s_l, i.e. ${}^p s_l = \{p_i | \{p_i, s_l\} \in E_S^P\}$
– p_i^s denotes the handlers connected to p_i, i.e. $p_i^s = \{s_l | \{p_i, s_l\} \in E_S^P\}$

As in the event net, the places in the intensity net contain tokens. These tokens can be used by the intensity handlers to produce intensities. A token is *active* if it is being used by an intensity handler, otherwise it is *idle*. While idle tokens are associated with places, active tokens are associated with edges. An intensity handler determines how much intensity is produced in its arcs as a function of the number of active tokens in its edges.

The state of the net is given by the variables associated with the net elements. Formally:

Definition 6 (State) *The state of an intensity net $\mathcal{N}_S$ is given by the tuple $(m, \mu_P, \mu_E, \Delta\lambda, \lambda)$, where:*

– $m \in \mathbb{R}_{\geq 0}^{|P|}$ *is the marking, i.e. a vector indexed by P where $m[p_i]$ is the number of tokens in p_i,*
– $\mu_P \in \mathbb{R}_{\geq 0}^{|P|}$ *is a vector indexed by P where $\mu_P[p_i]$ is the number of idle tokens in p_i,*
– $\mu_E \in \mathbb{R}_{\geq 0}^{|E_S^P|}$ *is a vector indexed by E_S^P where $\mu_E[\{p_i, s_l\}]$ is the number of active tokens of p_i being used by s_l,*
– $\Delta\lambda \in \mathbb{R}_{\geq 0}^{|E_S^T|}$ *is a vector indexed by E_S^T where $\Delta\lambda[(t_j, s_l)]$ is a decrease of intensity in t_j produced by s_l, and $\Delta\lambda[(s_l, t_j)]$ is an increase of intensity in t_j produced by s_l,*
– $\lambda \in \mathbb{R}_{\geq 0}^{|T|}$ *is a vector indexed by T where $\lambda[t_j]$ is the intensity in t_j.*

The number of tokens in a place p_i is equal to the number of its idle tokens plus the number of its active tokens:

$$m[p_i] = \mu_P[p_i] + \sum_{s_l \in p_i^s} \mu_E[\{p_i, s_l\}] \quad \forall\, p_i \in P \tag{11}$$

An intensity handler is said to be working when it is producing intensities. When an intensity handler $s_l \in S$ starts working, the number of idle tokens in ${}^p s_l$ decreases, the number of active tokens in its edges increases (such tokens start being used by the handler), and intensities are produced in its arcs. Conversely, when an intensity handler s_l stops working, the number of idle tokens in ${}^p s_l$ increases (i.e. they are released by the handler), the number of active tokens becomes 0, and no intensities are produced in its arcs. Thus (in contrast to the firing of event handlers whose firing cannot be reversed once it has occurred) intensity handlers are allowed to start and stop working (or to increase and decrease their working

rates), thus allocating tokens as active tokens and releasing them as idle tokens over time.

The relation between the number of active tokens and the intensities produced are given by a set of inequalities associated with each intensity handler $s_l \in S$. The coefficients of these inequalities can be captured by two matrices (C_l, D_l) of real numbers and same number of rows. The columns of C_l are indexed by the arcs connecting s_l to transitions. The columns of D_l are indexed by the edges connecting places to s_l. Matrix $C(D)$ is obtained by arranging all the matrices $C_l(D_l)$ *diagonally.*

Each transition t_j is assigned a default (or nominal) intensity $\lambda_0[t_j]$. Thus, the intensity $\lambda[t_j]$ in a transition t_j is equal to $\lambda_0[t_j]$ plus the positive changes in intensity minus the negative changes in intensity:

$$\lambda[t_j] = \lambda_0[t_j] - \sum_{s_l \in t_j^s} \Delta\lambda[(t_j, s_l)] + \sum_{s_l \in {}^s t_j} \Delta\lambda[(s_l, t_j)] \quad \forall\ t_j \in T \tag{12}$$

The number of active tokens, $\mu_w \in \mathbb{R}_{\geq 0}^{|^p s_l|}$ indexed by the edges of s_l, being used by an intensity handler s_l, and the intensities, $\Delta\lambda_w \in \mathbb{R}_{\geq 0}^{|^t s_l| + |s_l^t|}$ indexed by the arcs of s_l, produced by s_l are related by the matrices C_l and D_l as follows:

$$C_l \Delta\lambda_w \leq D_l \mu_w \tag{13}$$

If $\mathbf{1}\mu_w + \mathbf{1}\Delta\lambda_w > 0$, then s_l is said to be working. Similarly to event handlers, intensity handlers are not forced to work. When a number of intensity handlers work simultaneously, they share the tokens in places and collaborate in the production of intensities. In a similar way to (4), the inequalities (13) allow the modeling of uncertainty in the intensity changes produced by the active tokens.

Similarly to (7), the state equations of the intensity net determine the potential states of the net for a given marking m and default intensities λ_0:

Proposition 2 (State equations) *Let the state of an intensity net $\mathcal{N}_S$ be $(m, m, 0, 0, \lambda_0)$, i.e. m idle tokens are available and no intensity handler is working. Every state $(m, \mu_P, \mu_E, \Delta\lambda, \lambda)$ reachable from $(m, m, 0, 0, \lambda_0)$ belongs to $SE_{\mathcal{N}_S}(m, \lambda_0)$ where:*

$$\begin{aligned} SE_{\mathcal{N}_S}(m, \lambda_0) = \{(m, \mu_P, \mu_E, \Delta\lambda, \lambda) | \\ m = \mu_P + Y_m \mu_E \\ C\Delta\lambda \leq D\mu_E \\ \lambda = \lambda_0 + Z_\lambda \Delta\lambda\} \end{aligned} \tag{14}$$

where Y_m and Z_λ are matrices determined by the net structure:

- *Y_m is a matrix with rows indexed by P, columns indexed by E_S^P, and such that $Y_m[p_i, \{p_i, s_l\}] = 1\ \forall \{p_i, s_l\} \in E_S^P$ and the rest of the elements in Y_m are 0,*
- *Z_λ is a matrix with rows indexed by T, columns indexed by E_S^T, and such that $Z_\lambda[t_j, (t_j, s_l)] = -1\ \forall (t_j, s_l) \in E_S^T$, $Z_\lambda[t_j, (s_l, t_j)] = 1\ \forall (s_l, t_j) \in E_S^T$ and the rest of the elements in Z_λ are 0,*

and μ_P, μ_E, $\Delta\lambda$ and λ are nonnegative variables.

Similarly to (7), the equations $m{=}\mu_P{+}Y_m\mu_E$, $C\Delta\lambda{\leq}D\mu_E$, $\lambda{=}\lambda_0{+}Z_\lambda\Delta\lambda$ in (14) are the matrix forms of (11), (13) and (12) respectively, and thus, must hold at every state $(m, \mu_P, \mu_E, \Delta\lambda, \lambda)$ reachable from $(m, m, 0, 0, \lambda_0)$. As in (7), equations (14) account for the cumulative intensities produced by the handlers, and hence, $SE_{\mathcal{N}_S}(m, \lambda_0)$ can contain spurious solutions.

As in (8), inequalities can be considered to model linear relationships among the values of the default intensities, λ_0. Let us assume that λ_0 is a vector constrained as:

$$J_\lambda \lambda_0 \leq K_\lambda \tag{15}$$

where J_λ and K_λ are real matrices of appropriate size. Similarly to (8), the inequalities (15) can be used to model the uncertain default intensity of a transition, or to establish linear constraints among default intensities of transitions. Equations (14) can be easily modified to take into account the relationships expressed by (15):

$$\begin{aligned} SE_{\mathcal{N}_S}(m, J_\lambda, K_\lambda) = \{(m, \mu_P, \mu_E, \Delta\lambda, \lambda)| \\ m = \mu_P + Y_m\mu_E \\ C\Delta\lambda \leq D\mu_E \\ \lambda = \lambda_0 + Z_\lambda\Delta\lambda \\ J_\lambda\lambda_0 \leq K_\lambda\} \end{aligned} \tag{16}$$

Although intensity handlers are not forced to work, it is useful in some cases to consider only those states in which all the tokens of given places are active. Let $P_F \subseteq P$ be the set of places whose tokens must be active, i.e. the number of idle tokens of $p_i \in P_F$ must be 0. In order to constrain (16) to such a set of states, the following equation can be added:

$$\mu_P[p_i] = 0 \quad \forall\ p_i \in P_F \tag{17}$$

3.2 Modeling capabilities

Figure 4 shows some of the modeling capabilities of intensity nets. The default intensity, $\lambda_0[t]$, of a transition, t, can be written next to the transition, see Figure 4(b) (default intensities equal to 0 are omitted in the figures). As in the event nets, labels are associated with arcs and edges to represent amounts of produced/consumed intensities and number of active tokens.

The intensity net in Figure 4(a) has one place p_1, one transition t_1 and one intensity handler s_1. The inequality associated with s_1 establishes that the intensity $\Delta\lambda[(s_1, t_1)]$ produced in the arc (s_1, t_1) by s_1 must satisfy $2\mu_E[\{p_1, s_1\}]{\leq}\Delta\lambda[(s_1, t_1)]{\leq}3\mu_E[\{p_1, s_1\}]$ where $\mu_E[\{p_1, s_1\}]$ is the number of active tokens in $\{p_1, s_1\}$ (notice that given that $m[p_1]{=}2$, the number of active tokens is upper bounded by 2). The actual value of $\Delta\lambda[(s_1, t_1)]$ is selected nondeterministically in this interval. Since the default intensity of t_1 is 0 and (s_1, t_1) is the only arc connected to t_1, it holds $\lambda[t_1]{=}\Delta\lambda[(s_1, t_1)]$ what establishes the rate at which actions will be produced in t_1.

The intensity handler s_1 in Figure 4(b) makes use of the active tokens in p_1 to decrease the intensity in t_1 and increase the intensity in t_2. This can be seen as an

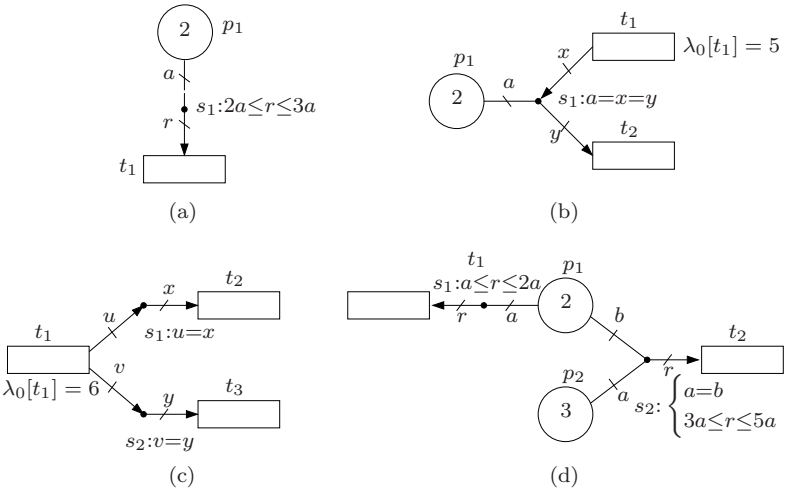


Fig. 4 Modeling capabilities of intensity nets.

intensity transfer from one transition to the other. According to the equations associated with s_1, which can be rewritten as $\mu_E[\{p_1, s_1\}] = \Delta\lambda[(t_1, s_1)] = \Delta\lambda[(s_1, t_2)]$, the amount of this transfer is equal to the number of active tokens, $\mu_E[\{p_1, s_1\}]$, which in this case is at most 2 given that $m[p_1] = 2$. Thus, if there is one active token, i.e. $\mu_E[\{p_1, s_1\}] = 1$, according to the state equations (14) the resulting intensities will be $\lambda[t_1] = \lambda_0[t_1] - \Delta\lambda[(t_1, s_1)] = \lambda_0[t_1] - \mu_E[\{p_1, s_1\}] = 5 - 1 = 4$ and $\lambda[t_2] = \lambda_0[t_2] + \Delta\lambda[(s_1, t_2)] = \lambda_0[t_2] + \mu_E[\{p_1, s_1\}] = 0 + 1 = 1$.

The net in Figure 4(c) shows how the intensity of one transition, t_1, can be used to produce intensity in other transitions, t_2 and t_3. For this net, the state equations (14) become $\lambda[t_1] = \lambda_0[t_1] - \Delta\lambda[(t_1, s_1)] - \Delta\lambda[(t_1, s_2)]$, $\lambda[t_2] = \lambda_0[t_2] + \Delta\lambda[(s_1, t_2)]$, $\lambda[t_3] = \lambda_0[t_3] + \Delta\lambda[(s_2, t_3)]$, $\Delta\lambda[(t_1, s_1)] = \Delta\lambda[(s_1, t_2)]$, $\Delta\lambda[(t_1, s_2)] = \Delta\lambda[(s_2, t_3)]$. Given that $\lambda_0[t_1] = 6$ and $\lambda_0[t_2] = \lambda_0[t_3] = 0$, these state equations reduce to $\lambda[t_1] + \lambda[t_2] + \lambda[t_3] = 6$ what summarizes the potential intensities of the net.

The net in Figure 4(d) models a choice in place p_1, i.e. each token in p_1 can be used either to produce an intensity within the interval $[1, 2]$ in t_1, or synchronize with a token in p_2 to produce an intensity within the interval $[3, 5]$ in t_2.

4 Flexible nets

This section introduces Flexible Nets (FNs), which can be denoted as PHT nets, i.e. places P and transitions T are connected by event and intensity handlers. Roughly, an FN consists of an event net and an intensity net that have the same set of places and the same set of transitions.

Definition 7 (Flexible net) *A Flexible Net (FN) is a tuple* $\mathcal{N} = (P, T, V, E_V, A, B, S, E_S, C, D)$ *where* (P, T, V, E_V, A, B) *is an event net and* (P, T, S, E_S, C, D) *is an intensity net.*

In an FN, the event net determines the way actions produce marking changes, and the intensity net determines the way tokens produce intensity changes. The inequalities associated with handlers allow the modeler to cover a range of relationships between "actions and tokens" and "tokens and intensities". Thus, handlers can be seen as a flexible layer between places and transitions that offers the possibility to model uncertainties in both the way actions produce marking changes, and the way tokens produce intensity changes.

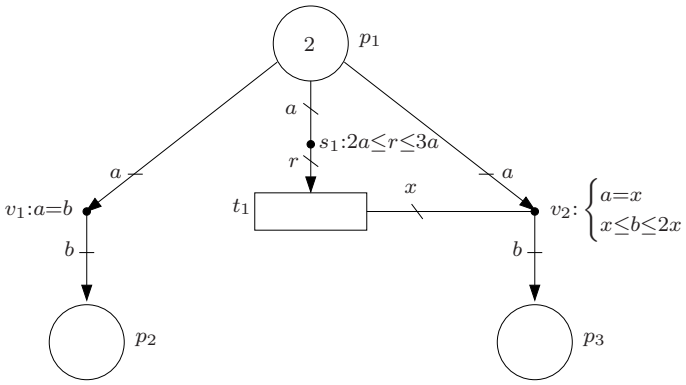


Fig. 5 FN resulting of combining the event net in Figure 2(a) and the intensity net in Figure 4(a).

The FN in Figure 5 is composed of the event net in Figure 2(a) and the intensity net in Figure 4(a). While the event net determines the marking changes produced by the firing of event handlers, the intensity net establishes the rate at which actions are created in t_1. Notice that the firing of v_2 implies the execution of actions in t_1, i.e. actions need to be produced in t_1 so that v_2 can fire. On the other hand, v_1 is not connected to any transitions and, thus, it can fire when there is a positive marking in p_1. It should be noted that this is not equivalent to an immediate transition in Petri nets, since the firing of t_2 is not forced to happen as soon as the marking of p_1 is positive, its firing can occur at any time at which the marking of p_1 is positive.

In order to compute the number of actions produced in transitions, the number of actions produced in the intensity arcs will be computed first. Let $\Delta\sigma(\tau)$ denote the number of actions produced in the intensity arcs until time τ ($\Delta\sigma[e](\tau)$ with $e \in E_S^T$ denotes the number of actions produced in e). The value of $\Delta\sigma(\tau)$ is defined as the integral of $\Delta\lambda$ over time:

$$\Delta\sigma(\tau) = \int_0^\tau \Delta\lambda(s)\,\mathrm{d}s \tag{18}$$

The overall number of actions, $\sigma[t_j](\tau)$, produced in a transition t_j can be computed by integrating $\lambda[t_j]$, or equivalently, by making use of Z_λ, see (14), and $\Delta\sigma(\tau)$:

$$\sigma(\tau) = \lambda_0\tau + Z_\lambda\Delta\sigma(\tau) \tag{19}$$

In addition to the state variables of the event and intensity net, $\Delta\sigma$ is included in the tuple of variables defining the state of the FN.

Definition 8 (State) *The state* $\mathbf{x}$ *of an FN is given by the tuple* $\mathbf{x} = (m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$.

All the state variables are time dependent. For the sake of clarity, the time dependency will be omitted when it is clear from the context, e.g. $m(\tau)$ is shortened to m. At time 0 it holds $\Delta\sigma = 0$, $\sigma = 0$, $a_T = 0$, $a_E = 0$, $\Delta m = 0$, i.e. the initial state can be written as: $(m, \mu_P, \mu_E, \Delta\lambda, \lambda, 0, 0, 0, 0, 0)$.

By making use of $SE_{\mathcal{N}_V}(\sigma, J_m, K_m)$ in (9), $SE_{\mathcal{N}_S}(m, J_\lambda, K_\lambda)$ in (16), (18) and (19), it is possible to write a set of equations that any potential state at time τ must satisfy.

Proposition 3 (State equations) *Let* $\mathcal{N}$ *be an FN with initial marking* m_0 *satisfying* $J_m m_0 \leq K_m$, *and default intensities* λ_0 *satisfying* $J_\lambda \lambda_0 \leq K_\lambda$. *Every state* $(m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$ *reachable at time* τ *belongs to* $SE_{\mathcal{N}}(\tau, J_m, K_m, J_\lambda, K_\lambda)$ *where:*

$$\begin{aligned} SE_{\mathcal{N}}(\tau, J_m, K_m, J_\lambda, K_\lambda) = \{(m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m) \mid \\ m = \mu_P + Y_m \mu_E;\ C\Delta\lambda \leq D\mu_E;\ \lambda = \lambda_0 + Z_\lambda \Delta\lambda; J_\lambda \lambda_0 \leq K_\lambda \\ \Delta\sigma = \int_0^\tau \Delta\lambda(s)\,\mathrm{d}s;\ \sigma = \lambda_0 \tau + Z_\lambda \Delta\sigma \\ \sigma = a_T + Y_\sigma a_E;\ A\Delta m \leq B a_E;\ m = m_0 + Z_m \Delta m;\ J_m m_0 \leq K_m\} \end{aligned} \tag{20}$$

where every variable is nonnegative.

This way, an FN is a continuous time model where time, denoted as τ, is the independent variable and all the state variables are nonnegative reals.

Equations (20) can be interpreted as follows: at a given time τ, some of the produced actions (σ) are available (a_T), and the rest (a_E) were executed before τ. The executed actions produced marking changes (Δm) which resulted in the marking m in places at τ. Some of the tokens in m are active (μ_E) and the rest are idle (μ_P). Active tokens produce intensity changes ($\Delta\lambda$) which result in overall intensities (λ) in transitions at τ. The integral of the intensity changes and overall intensities over time after τ will produce more actions (σ), i.e. σ is produced as time elapses. This behavior repeats over time: when a new marking is reached, intensities are updated, which can lead to the production and execution of new actions, which consequently results in a new marking.

As in the event and intensity nets, constraints (10) and (17) can be added to (20) to force the execution of actions and the activity of places.

5 Exploiting uncertainty

The state equations (20) account for all the potential states of the net at time τ. In order to facilitate the analysis of FNs, a set of necessary reachability conditions was developed [17]. These conditions consist of linear and quadratic inequalities that all the solutions of (20) must satisfy during the interval $[0, \tau]$. In order to obtain a time trajectory of the state, i.e. values of the state at different time instants τ_1, τ_2, τ_3, ..., two methods are considered:

1. The first method consists of developing a unique set of necessary reachability conditions that combines the reachability conditions of each interval $[0, \tau_1], [\tau_1, \tau_2], [\tau_2, \tau_3], \ldots$, in such a way that all the states that satisfy the constraints at the end of a given interval are taken as potential initial states for the next interval (see [17] for details). Once this set of constraints is obtained, a particular trajectory of the FN can be computed by adding an objective function to such a set of constraints, and by solving the resulting programming problem.
2. The second method follows a model predictive control (MPC) [18] approach. According to this approach, the programming problem described in the first method is solved over the intervals $[0, \tau_1], [\tau_1, \tau_2], \ldots, [\tau_{n-1}, \tau_n]$ where τ_n is the prediction horizon. Then, the state obtained at τ_1 is taken as the initial state and a programming problem over the intervals $[\tau_1, \tau_2], [\tau_2, \tau_3], \ldots, [\tau_n, \tau_{n+1}]$ is defined and solved. This procedure can be repeated with the subsequent intervals.

It should be noted that solving convex quadratic programming problems is required by both methods above. Given that the computational complexity required to solve such problems is polynomial, the proposed computational methods can be applied to large FNs. The following subsections present some of the modeling, analysis and control capabilities of FNs by modeling a linear system with uncertain parameters, a resource allocation system and a system with control actions.

5.1 Linear system with uncertain parameters

The FN in Figure 6 models a linear system with uncertain dynamics. More precisely, if we assume that all the tokens are forced to be active and all the actions to be executed, the rate at which the marking changes can be expressed as:

$$\dot{m}[p_1] = -q - hm[p_1] + m[p_2] + 2 \tag{21}$$

$$\dot{m}[p_2] = q - m[p_2] \tag{22}$$

$$\dot{m}[p_3] = hm[p_1] - 2 \tag{23}$$

where q and h are uncertain parameters but known to be in the intervals: $q \in [1.0, 1.5]$, $h \in [0.9, 1.1]$. Thus, any potential time trajectory of the system will satisfy (21) with values of q and h within the given intervals. The uncertain parameter q is modeled by the default intensity of t_1, and h is modeled by the inequalities of s_3. Notice that the FN in Figure 6 combines transitions with constant speed, e.g. t_4, transitions whose speed is proportional to the marking of a place, e.g. t_2, and uncertain parameters.

Let the initial marking be $m_0[p_1]$=4, $m_0[p_2]$=0, and $m_0[p_3]$=0. Figure 7 shows the time trajectories of the marking and the intensities of t_1 and t_3 under different objective functions (notice that the intensity of t_2 is equal to $m[p_2]$, and the intensity of t_4 is constant and equal to 2). The trajectories have been obtained by an MPC approach with a sample time (or interval) of 0.1 time units and a prediction horizon of one sample time. This means that initially, i.e. at time 0.0, the programming problem [17] is defined and solved over the time interval [0.0, 0.1]. The solution of the problem is taken as the state of the system at time 0.1. Then,

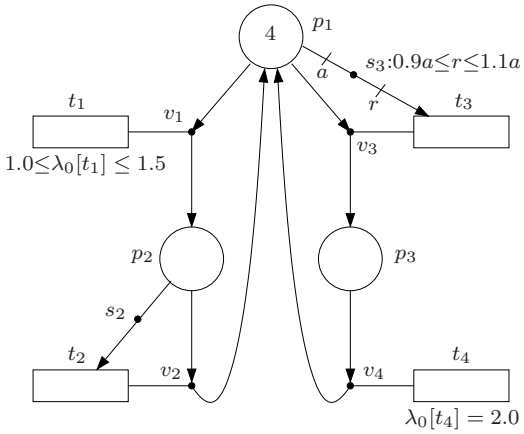


Fig. 6 FN with uncertain dynamics modeled by the default intensity of t_1 and the inequalities of s_3.

the programming problem is defined and solved over the interval [0.1, 0.2], and the procedure is repeated.

The trajectory in Figure 7(a) is obtained by the objective function "$min\ m[p_3]$", i.e. the goal is to minimize the marking of p_3 at the end of each interval. In the plots, $\bar{\lambda}[t_j]$ denotes the average intensity of t_j during each interval. For such an objective, the solution of the programming problem sets the uncertain parameters to $\lambda_0[t_1]$=1.5 and $0.9a$=r (which results in $\lambda[t_3]$=$0.9m[p_1]$). This setting minimizes the flow directed from p_1 to the branch composed of v_3 and v_4. As expected, the intensity of t_1 and t_2 (t_3 and t_4) is the same at steady state.

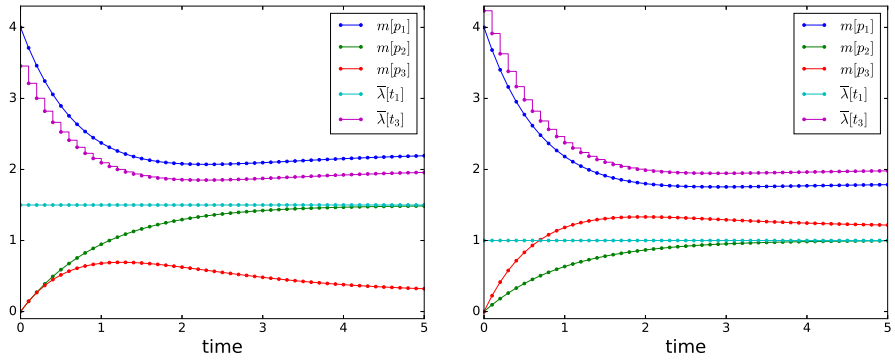


Fig. 7 Time evolution of the FN in Figure 6 when $m[p_3]$ is minimized (a) and maximized (b).

The trajectory in Figure 7(b) is obtained by the objective function "$max\ m[p_3]$". For such an objective, the solution of the programming problem sets $\lambda_0[t_1]$=1.0 and r=$1.1a$ (which results in $\lambda[t_3] = 1.1m[p_1]$). This setting maximizes the flow directed from p_1 to the branch composed of v_3 and v_4.

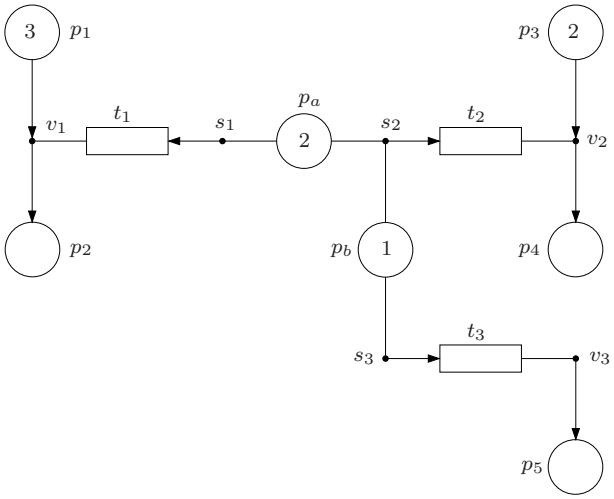
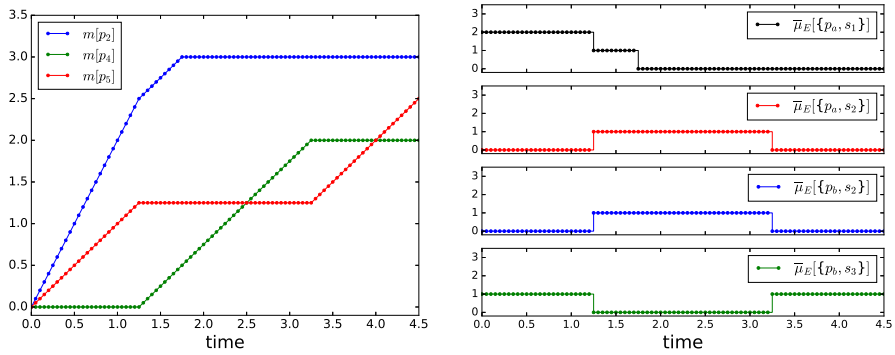


Fig. 8 FN modeling a resource allocation system with three production lines and two shared resources.

5.2 Resource allocation

The FNs in Figure 8 models a dynamic system in which shared resources can be allocated to different production lines. Such a net shows how the tokens of a given place can activate different processes (those places have several intensity edges) and can cooperate with active tokens of other places. Namely, there are two types of resources, p_a and p_b, and three production lines, t_1, t_2 and t_3. The production line associated with $t_1(t_2)$ uses the raw material modeled by the tokens in $p_1(p_3)$ and produces items modeled by the tokens in $p_2(p_4)$. The production line associated with t_3 produces tokens in p_5 and it is assumed that it requires no raw material (or equivalently, this raw material is inexhaustible). In order to operate, the production line associated with $t_1(t_3)$ requires the allocation of resources of type $p_a(p_b)$. The speed of these production lines, t_1 and t_3, is proportional to the number of tokens allocated to them. The operation of production line t_2 requires the cooperation of both resources, p_a and p_b, i.e. tokens of both resources must synchronize in equal amounts to make t_2 work. The speed of t_2 is equal to the number of tokens of p_a (or p_b) allocated to this production line.

While the event handlers, v_1, v_2 and v_3 determine the relationship between the input and output material of the production lines, the intensity handlers specify the speed of these lines according to the number of active tokens assigned to each line. In particular, the intensity edges $\{p_a, s_1\}$ and $\{p_a, s_2\}$ model the fact that the active tokens of p_a can be used either by s_1 or s_2. In a similar way, the fact that the active tokens of p_b can be used either by s_2 or s_3 is modeled by the intensity edges $\{p_b, s_2\}$ and $\{p_b, s_3\}$. This way, s_2 is the intensity handler responsible for the synchronization of resources for t_2. The actions of all transitions are forced to be executed in order to model the active tokens to make the production lines work. Note that the graphical representation of the system by an FN is reasonably clear and compact. If the system were modeled by a classical Petri net, each transition would have to be split into several transitions that would each model



(a) (b)

Fig. 9 Time trajectories of the marking (a) of the net in Figure 8, and of the number of resources allocated to the transitions (b).

the acquisition and release of the resources and the speed of the production lines. This would lead to a more complex graphical notation and, potentially, to more involved analysis methods.

Let the initial marking of the net be $m_0[p_1]$=3, $m_0[p_2]$=0, $m_0[p_3]$=2, $m_0[p_4]$=0, $m_0[p_5]$=0, $m_0[p_a]$=2 and $m_0[p_b]$=1, i.e. there are two copies of resource type p_a and one copy of resource type p_b. Assume that the goal is to compute how the resources must be allocated over time so that the objective function $\bar{m}[p_2]$+0.5$\bar{m}[p_4]$+0.25$\bar{m}[p_5]$, where $\bar{m}[p_i]$ denotes the average marking of p_i, is maximized. In words, this objective function implies that the goal is to maximize the production of all items giving priority to the products of type p_2, then p_4 and finally p_5.

This resource allocation problem can be solved by a single programming problem (see first method in Section 5) that makes use of the reachability constraints in [17] and the mentioned objective function. More precisely, in order to obtain time trajectories, 90 intervals, each of 0.05 time units, will be considered.

The time trajectories of the marking and the allocated resources are shown in Figure 9(a) and 9(b) respectively. Four time periods with different resource allocations (or operation modes) can be distinguished in these figures. The first period, from time 0 to 1.25, allocates the two tokens of p_a to s_1. This gives a high yield in the production of the items in p_2 which has the highest priority. Given that the two tokens of p_a are used by s_1 during this first period, the token in p_b cannot be used by s_2, and hence it is used by s_3 to produce the items in p_5, which has the lowest priority. During the second time period, from time 1.25 to 1.75, one token of p_a is used by s_1, and the other token of p_a is synchronized by s_2 with the token of p_b to operate t_2 and produce the items in p_4, which has medium priority. As a result, the speed of $t_1(t_2)(t_3)$ is 1(1)(0) during the second time period. At time 1.75, the marking of p_1 becomes 0, and hence, the active token of p_a allocated to s_1 is released and becomes idle. Thus, during the third period, from time 1.75 to 3.25, only t_2 is working. At time 3.25, the marking of p_3 becomes 0, and hence the two tokens of p_a become idle. During the fourth period, from time 3.25 onward, only the token of p_b is active, and is employed by s_3 to operate t_3.

5.3 Control actions

Control actions can be modeled in FNs by means of default intensities. This section demonstrates the ability of FNs to model and solve a control problem in which the control action is dynamically constrained.

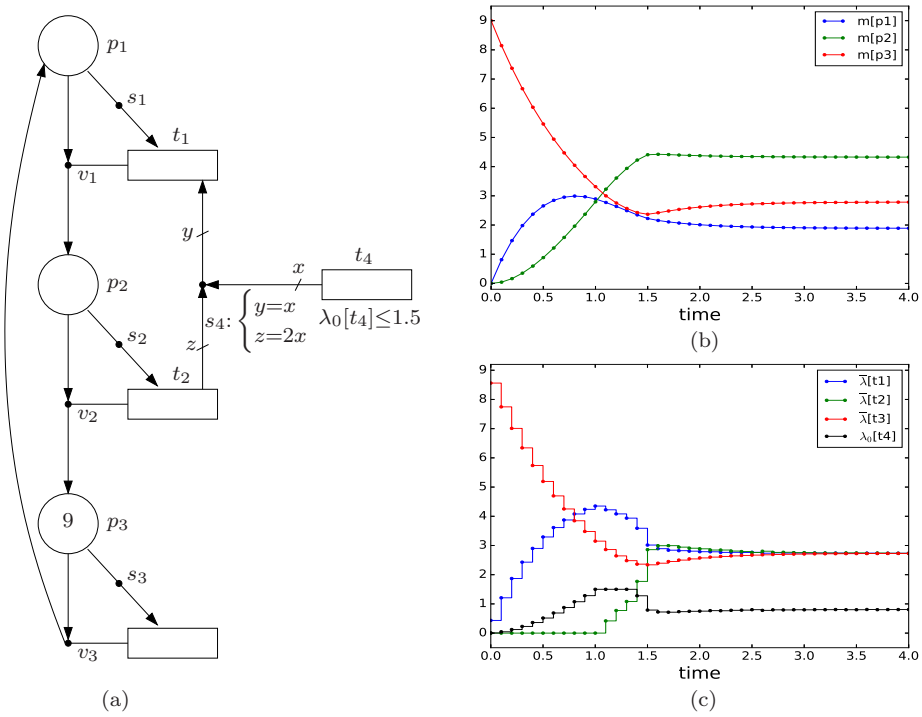


Fig. 10 (a) FN with control action modeled by $\lambda_0[t_4]$. (b) Time evolution of the marking. (c) Time evolution of the average intensities during each interval, $\bar{\lambda}[t_1]$, $\bar{\lambda}[t_2]$, $\bar{\lambda}[t_3]$, and control action $\lambda_0[t_4]$.

Figure 10(a) depicts a net with three places and four transitions. All the tokens are forced to be active, and all the actions are forced to be executed. The initial marking is $m_0[p_1]{=}m_0[p_2]{=}0$ and $m_0[p_3]{=}9$. The default intensities of t_1, t_2 and t_3 are 0. The default intensity of t_4, $\lambda_0[t_4]$, models the only control action that can be applied to the system, and is constrained to the interval $[0, 1.5]$. Given that the equations associated with s_4 are $s_4{:}y{=}x; z{=}2x$, each intensity unit in t_4 increases the intensity in t_1, $\lambda[t_1]$, by one unit and decreases the intensity in t_2, $\lambda[t_2]$, by two units. This way, the same control action is used for the intensities of t_1 and t_2. Thus, the intensities in transitions satisfy $\lambda[t_1]{=}m[p_1]{+}\lambda_0[t_4]$, $\lambda[t_2]{=}m[p_2]{-}2\lambda_0[t_4]$, $\lambda[t_3]{=}m[p_3]$. Notice that the input action $\lambda_0[t_4]$ is not only statically constrained by $\lambda_0[t_4]{\leq}1.5$, but also dynamically constrained by $\lambda_0[t_4]{\leq}0.5m[p_2]$ (if this constraint is violated then $\lambda[t_2]$ becomes negative).

The evolution of the system can be described by the following differential equations:

$$\dot{m}[p_1] = \lambda[t_3] - \lambda[t_1] = m[p_3] - m[p_1] - \lambda_0[t_4] \tag{24}$$

$$\dot{m}[p_2] = \lambda[t_1] - \lambda[t_2] = m[p_1] - m[p_2] + 3\lambda_0[t_4] \tag{25}$$

$$\dot{m}[p_3] = \lambda[t_2] - \lambda[t_3] = m[p_2] - 2\lambda_0[t_4] - m[p_3] \tag{26}$$

where all the variables are nonnegative and $\lambda_0[t_4] \leq 1.5$.

Consider the objective function $min\ (m[p_1] - 1)^2 + (m[p_2] - 4)^2$. Notice that in this system the invariant $m[p_1] + m[p_2] + m[p_3] = 9$ holds, then, the control objective is to drive the system to a marking that is as close as possible to the target marking $(1, 4, 4)$. Figures 10(b) and (c) show the trajectories obtained by MPC with a sample time of 0.1 time units and a prediction horizon of one step. Initially, the value of $\lambda_0[t_4]$ is low as it is constrained by $m[p_2]$, which initially is 0. Then, $\lambda_0[t_4]$ increases so that $m[p_2]$ increases and $m[p_1]$ decreases. At time 1.0, $\lambda_0[t_4]$ hits the constraint 1.5 where it is kept constant for 0.4 time units. Then, $\lambda_0[t_4]$ decreases in order to approach further the target marking. At steady state, the average intensities of t_1, t_2 and t_3 are the same and equal to 2.73, and the value of the control action is $\lambda_0[t_4] = 0.81$. The steady state marking reached is $(1.93, 4.35, 2.73)$. It is important to note that the target marking $(1, 4, 4)$ cannot be an achievable steady state marking with the proposed single control action.

All the trajectories in this paper have been obtained by the tool `fnyzer` (`https://bitbucket.org/Julvez/fnyzer.git`). This tool makes use of the modeling language Pyomo [14,15] and solvers, such as Gurobi [13] and CPLEX [1], to solve the programming problems associated with the FNs. The CPU time (Intel i7, 2.00 GHz, 8 GiB, Ubuntu 14.04 LTS) to solve one step of the MPC approach for the FNs in Figures 6 and 10 was $1.81s$ and $5.93s$ respectively. The CPU time to solve the only programming problem associated with Figure 8 was $1.27s$.

6 Conclusions

FNs consist of two nets, an event net and an intensity net, that make an explicit distinction between the parts of the system involved in updating the marking in places, i.e. the event net, and the parts of the system involved in the determination of the speeds of transitions, i.e. the intensity net. Both the event and the intensity net are tripartite graphs in which places and transitions are connected by event and intensity handlers, respectively. This way, handlers act as an intermediate layer between places and transitions, which results in a significant modeling power. For instance, a transition in an event net can consume tokens from different sets of places, and a place in an intensity net can regulate the speed of different transitions. The tripartite net structure of event and intensity nets has demonstrated to be useful to model partial observability and resource allocation.

Different types of system uncertainties can be accommodated by FNs through sets of linear inequalities associated with places, transitions, event handlers and intensity handlers. Namely, these inequalities allow the modeling of uncertainty in: a) the initial marking (8); b) the default intensities (15); c) the marking change produced by the execution of actions (4); and d) the intensity change produced by the active tokens (13). FNs account for the potential system trajectories arising as

a result of uncertainties by means of a set of constraints that represent necessary reachability conditions. The combination of these constraints with an objective function can be used to obtain a system trajectory that optimizes a given criterion. This approach was successfully used to compute trajectory bounds, for instance, in the presented linear system with uncertain parameters, or to obtain a control law in a system whose control action is modeled by a transition with uncertain default intensity.

Acknowledgements

This work was supported by the European Commission through a 7^{th} Framework Program BIOLEDGE Contract No: 289126 to SGO, and a Marie Curie Intra European Fellowship to JJ (FormalBio Contract No: 623995, Call reference: FP7-PEOPLE-2013-IEF). Further support came from the Biotechnology & Biological Sciences Research Council (UK) grant no. BB/N02348X/1 as part of the IBiotech Program, and by the Industrial Biotechnology Catalyst (Innovate UK, BBSRC, EPSRC) to support the translation, development and commercialisation of innovative Industrial Biotechnology processes.

References

1. IBM ILOG CPLEX Optimizer, 2010.
2. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.
3. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, Apr. 1994.
4. N. Bertrand, P. Bouyer, Th. Brihaye, Q. Menet, C. Baier, M. Größer, and M. Jurdziński. Stochastic timed automata. *Logical Methods in Computer Science*, 10(4:6), Dec. 2014.
5. L. Bortolussi, J. Hillston, D. Latella, and M. Massink. Continuous approximation of collective system behaviour: A tutorial. *Performance Evaluation*, 70(5):317 – 349, 2013.
6. W. Borutzky. *Bond Graph Methodology – Development and Analysis of Multidisciplinay Dynamic System Models*. Springer-Verlag, London, UK, 2010. ISBN : 978-1-84882-881-0.
7. M. Braun, W. Lucas, C. Coleman, and D. Drew. *Differential equation models*. Modules in Applied Mathematics. Springer-Verlag, 1983.
8. M. P. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9):1531 – 1539, 2010.
9. J. Cardoso, R. Valette, and D. Dubois. Possibilistic Petri nets. 29:573–82, 02 1999.
10. F. Ciocchetta and J. Hillston. Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks. *Electron. Notes Theor. Comput. Sci.*, 194(3):103–117, Jan. 2008.
11. A. Clark, S. Gilmore, J. Hillston, and M. Tribastone. *Stochastic Process Algebras*, pages 132–179. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
12. A. Detwarasiti and R. D. Shachter. Influence Diagrams for Team Decision Analysis. *Decision Analysis*, 2(4):207–228, 2005.
13. Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual, 2015.
14. W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo–optimization modeling in Python*, volume 67. Springer Science & Business Media, second edition, 2017.
15. W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, 3(3):219–260, 2011.
16. E. Jiménez, J. Júlvez, L. Recalde, and M. Silva. Relaxed continuous views of discrete event systems: considerations on Forrester diagrams and Petri nets. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 5, pages 4897–4904 vol.5, Oct 2004.

17. J. Júlvez, D. Dikicioglu, and S. G. Oliver. Handling variability and incompleteness of biological data by flexible nets: a case study for Wilson disease. *npj Systems Biology and Applications*, 4(1):7, 1 2018.
18. B. Kouvaritakis and M. Cannon. *Model Predictive Control. Classical, Robust and Stochastic*. Springer International Publishing, 2016.
19. C. G. Looney. Fuzzy Petri nets for rule-based decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):178–183, Jan 1988.
20. P. Merlin and D. J. Faber. Recoverability of communication protocols. *IEEE Trans. on Communications*, 24(9):1036–1043, 1976.
21. T. Murata. Petri Nets: Properties, Analysis and Applications. *Procs. of the IEEE*, 77(4):541–580, 1989.
22. C. J. Needham, J. R. Bradford, A. J. Bulpitt, and D. R. Westhead. A Primer on Learning in Bayesian Networks for Computational Biology. *PLOS Computational Biology*, 3(8):1–8, 08 2007.
23. J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, and B. Ø. Palsson. A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011. *Molecular Systems Biology*, 7(1), 2011.
24. C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25 – 31, 2001. Process Algebra.
25. A. Ramirez-Trevino, I. Rivera-Rangel, and E. Lopez-Mellado. Observability of discrete event systems modeled by interpreted Petri nets. *IEEE Transactions on Robotics and Automation*, 19(4):557–565, Aug 2003.
26. I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang. Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.
27. M. Silva, J. Júlvez, C. Mahulea, and C. R. Vázquez. On fluidization of discrete event models: observation and control of continuous Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 21(4):427–497, 2011.
28. M. Silva, E. Teruel, and J. M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Net Systems. *Lecture Notes in Computer Science*, 1491:309–373, 1998.
29. J. J. Tyson, W. T. Baumann, C. Chen, A. Verdugo, I. Tavassoly, Y. Wang, L. M. Weiner, and R. Clarke. Dynamic modelling of oestrogen signalling and cell fate in breast cancer cells. *Nature reviews. Cancer*, 11 7:523–32, 2011.
30. R. van den Berg, E. Lefeber, and K. Rooda. Modeling and control of a manufacturing flow line using partial differential equations. *Control Systems Technology, IEEE Transactions on*, 16:130 – 136, 02 2008.
31. A. Varma and B. Ø. Palsson. Metabolic Flux Balancing: Basic Concepts, Scientific and Practical Use. *Nature Biotechnology*, 12(10):994–998, Oct. 1994.
32. R. S. Wang, A. Saadatpour, and R. Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical Biology*, 9(5):055001, 2012.